## Parallelism Rule and Disjoint Conditions

In the previous lecture, we have seen Sequentialization rule is not easy to use when there are many threads, we need to calculate a lot of pre- or post- conditions to prove sequentialization.

Let's come up with some rules of inference that are easy to use. Someone suggested that, using the fact that disjoint threads can write at the same time without affecting variables in other threads, we come up with the parallelism rule.

- **Parallelism Rule (v 1.0)?**
  If $S_1, S_2, \dots, S_n$ are pairwise disjoint:
  1 $\{p_1\}\, S_1\, \{q_1\}$
  2 $\{p_2\}\, S_2\, \{q_2\}$
  …
  n $\{p_n\}\, S_n\, \{q_n\}$
  $\mathrm{n}+1$ $\{p_1 \wedge p_2 \wedge \dots \wedge p_n\}\, [S_1 \parallel S_2 \parallel \cdots \parallel S_n]\, \{q_1 \wedge q_2 \wedge \dots \wedge q_n\}$ parallelism $1, 2, \dots, \mathrm{n}$

- In the format of proof outline, we write: $\{p_1 \wedge p_2 \wedge \dots \wedge p_n\}\, [\{p_1\}\, S_1\, \{q_1\} \parallel \{p_2\}\, S_2\, \{q_2\} \parallel \cdots \parallel \{p_n\}\, S_n\, \{q_n\}]\, \{q_1 \wedge q_2 \wedge \dots \wedge q_n\}$

1. Prove $\{x = y = z = c\}\, [x := x^2 \parallel y := y^2 \parallel z := (z - d) * (z + d)]\, \{x = y = z + d^2\}$, where $c, d$ are named constants.
   First, we need to prove that all threads are pairwise disjoint.

| $i$ | $j$ | $vars(S_i)$ | $changes(S_j)$ | $S_j$ interferes with $S_i$? |
|---|---|---|---|---|
| 1 | 2 | $x$ | $y$ | $No$ |
| 2 | 1 | $y$ | $x$ | $No$ |
| 1 | 3 | $x$ | $z$ | $No$ |
| 3 | 1 | $z$ | $x$ | $No$ |
| 2 | 3 | $y$ | $z$ | $No$ |
| 3 | 2 | $z$ | $y$ | $No$ |

Using parallelism rule, we can come up with the following full proof outline:

$\{x = y = z = c\}\, \{x = c \wedge y = c \wedge z = c\}$
$[\{x = c\}\, x := x^2\, \{x_0 = c \wedge x = x_0^2\}\, \{x = c^2\}$
$\parallel \{y = c\}\, y := y^2 \{y_0 = c \wedge y = y_0^2\}\, \{y = c^2\}$
$\parallel \{z = c\}\, z := (z - d) * (z + d)\, \{z_0 = c \wedge z = (z_0 - d) * (z_0 + d)\}\, \{z = c^2 - d^2\}\,]$
$\{x = c^2 \wedge y = c^2 \wedge z = c^2 - d^2\}$
$\{x = y = z + d^2\}$

- The parallelism rule (v 1.0) is not correct, it can fail in some disjoint parallel programs.

2. For each of the following proof outlines, point out what is wrong in the proof and why parallelism rule (v 1.0) does not work.
   - $\{x = 0\}[\{x = 0\}\, x := 1\, \{x = 1\} \parallel \{x = 0\}\, y := 0\, \{x = y = 0\}]\, \{x = 1 \wedge x = y = 0\}$

    - The postcondition is wrong: $x = 1 \land x = y = 0$ is false, but this is not the case if we analyze the program.
    - The assignment $x \coloneqq 1$ in the first thread can make the precondition $x = 0$ of the second thread not true anymore.
    - The assignment $x \coloneqq 1$ also interferes with the postcondition $x = y = 0$ of the second thread, because the $x$ in this postcondition represents the $x$ that's before assignment $x \coloneqq 1$.

  - $\{x \geq y \land y = z\}$ $[\{x \geq y\}\ x \coloneqq x + 1\ \{x > y\} \parallel \{y = z\}\ y \coloneqq y * 2; z \coloneqq z * 2\ \{y = z\}]$ $\{x > y \land y = z\}$
    - The postcondition is wrong: we don't always have $x > y$ in the postcondition. For example, if we execute the program in a state where $x, y, z$ all evaluated to 2, then we will end up with $x$ updated to 3, and $y, z$ both updated to 4.

    - The assignment $y \coloneqq y * 2$ in the second thread can make the precondition $x \geq y$ of the first thread not true anymore.
    - The assignment $y \coloneqq y * 2$ also interferes with the postcondition $x > y$ of the second thread, because the $y$ in this postcondition represents the $y$ that's before assignment $y \coloneqq y * 2$ .

- We have the following observation. So that parallelism rule can work in a disjoint parallel program, we need more: "a thread doesn't change variables appear in the pre- and post- conditions of other threads".

- For a set of predicates $p_1, p_2, \dots, p_n$, we define $free(p_1, \dots, p_n) =$ the set of variables that are free (have free occurrences) in $p_1, p_2, \dots, p_n$.
- Given triples $\{p_1\}\ S_1\ \{q_1\}$ and $\{p_2\}\ S_2\ \{q_2\}$, we say **$S_1$ interferes with the conditions of $S_2$** if $change(S_1) \cap free(p_2, q_2) \neq \emptyset$. And we say **$S_1$ and $S_2$ have disjoint conditions** if $change(S_1) \cap free(p_2, q_2) = \emptyset$ and $change(S_2) \cap free(p_1, q_1) = \emptyset$.

Here, we give the real parallelism rule.

- Parallelism Rule
  If $S_1, S_2, \dots, S_n$ are pairwise disjoint, and $S_1, S_2, \dots, S_n$ have disjoint conditions:
  1 $\{p_1\}\ S_1\ \{q_1\}$
  2 $\{p_2\}\ S_2\ \{q_2\}$
  …
  n $\{p_n\}\ S_1\ \{q_n\}$
  n + 1 $\{p_1 \land p_2 \land \dots \land p_n\}\ [S_1 \parallel S_2 \parallel \cdots \parallel S_n]\ \{q_1 \land q_2 \land \dots \land q_n\}$ parallelism 1,2, … , n

3. Are the following threads pairwise disjoint? Do they have disjoint conditions?
   $\{p_1\}\ S_1\ \{q_1\} \equiv \{x \geq y\}\ x \coloneqq x + 1\ \{x > y\}$
   $\{p_2\}\ S_2\ \{q_2\} \equiv \{y = z\}\ y \coloneqq y * 2; z \coloneqq z * 2\ \{y = z\}$

   We can come up with the following table:

| $i$ | $j$ | $change(S_i)$ | $vars(S_j)$ | $free(p_j, q_j)$ | $S_i\ intf\ S_j$ | $S_i\ intf\ cond_j$ |
|---|---|---|---|---|---|---|
| 1 | 2 | $x$ | $y, z$ | $y, z$ | $No$ | $No$ |
| 2 | 1 | $y, z$ | $x$ | $x, y$ | $No$ | $Yes$ |

   So, $S_1$ and $S_2$ are disjoint, but they don't have disjoint conditions. That's why we cannot guarantee that we can use the parallelism rule to prove $\{p_1 \land p_2\}\ [S_1 \parallel S_2]\ \{q_1 \land q_2\}$.

## Removing Interference of Conditions for Disjoint Parallel Programs

What if we want to find a way to find a way to prove disjoint parallel programs using the parallelism rule? We want to find a way to remove the interference of conditions among disjoint threads. This can be done by **aging all the free variables in the precondition.**

4. Find some $q$, and use the parallelism rule to prove the following parallel program.

$$\{x = 0\}[\ \{x = 0\}\ x \coloneqq 1\ \{x = 1\} \parallel \{x = 0\}\ y \coloneqq 0\ \{x = y = 0\}\ ]\ \{q\}$$

- $S_1$ interferes the conditions of $S_2$
  - The assignment $x \coloneqq 1$ can make the precondition $x = 0$ not true anymore.
  - The assignment $x \coloneqq 1$ also interferes with the postcondition $x = y = 0$, because the $x$ in this postcondition represents the $x$ that's before assignment $x \coloneqq 1$.
- To avoid this interference, we can age all the $x$ in preconditions.

- For the first thread: $\{x_0 = 0 \land x = x_0\}\ x \coloneqq 1\ \{x_0 = 0 \land x = 1\}\ \{x = 1\}$
- For the second thread: $\{x_0 = 0\}\ y \coloneqq 0\ \{x_0 = 0 \land y = 0\}$

Thus, we can come up with the following full proof outline that uses parallelism rule:
$\{x_0 = 0 \land x = x_0\}$
$[\ \{x_0 = 0 \land x = x_0\}\ x \coloneqq 1\ \{x_0 = 0 \land x = 1\}\ \{x = 1\} \parallel$
$\{x_0 = 0\}\ y \coloneqq 0\ \{x_0 = 0 \land y = 0\}]$
$\{x_0 = 0 \land x = 1 \land y = 0\}$

5. Find some $q$, and use the parallelism rule to prove the following parallel program.

$$\{x \ge y \land y = z\}\ [\{x \ge y\}\ x \coloneqq x + 1\ \{x > y\} \parallel \{y = z\}\ y \coloneqq y * 2; z \coloneqq z * 2\ \{y = z\}]\ \{q\}$$

- We know that the assignment $y \coloneqq y * 2$ interferes with the pre- and post- conditions of the first thread; and we can age $y$ to avoid this interference. However, **aging all the free variables in preconditions** is a simple trick.
  For every free variable $x$ in the precondition of a thread:
  - If other thread writes it, then aging $x$ can remove interference in conditions.
  - If the current thread writes it, then aging $x$ is needed for forward assignment.
  - If no thread writes $x$, then $x$ is not updated anywhere and aging it is safe.

- For the first thread: $\{x_0 \ge y_0 \land x = x_0\}\ x \coloneqq x + 1\ \{\ x_0 \ge y_0 \land x = x_0 + 1\}\ \{x > y_0\}$.
- For the second thread: $\{y_0 = z_0 \land y = y_0 \land z = z_0\}\ y \coloneqq y * 2; \{y_0 = z_0 \land y = y_0 * 2 \land z = z_0\}\ z \coloneqq z * 2\ \{y_0 = z_0 \land y = y_0 * 2 \land z = z_0 * 2\}\ \{y = z\}$.

Thus, we can come up with the following full proof outline that uses parallelism rule:

$\{x_0 \ge y_0 \land y_0 = z_0 \land x = x_0 \land y = y_0 \land z = z_0\}$
$[\ \{\ x_0 \ge y_0 \land x = x_0\}\ x \coloneqq x + 1\ \{\ x_0 \ge y_0 \land x = x_0 + 1\}\{x > y_0\} \parallel$
$\{y_0 = z_0 \land y = y_0 \land z = z_0\}\ y \coloneqq y * 2; \{y_0 = z_0 \land y = y_0 * 2 \land z = z_0\}$
$\quad z \coloneqq z * 2\ \{y_0 = z_0 \land y = y_0 * 2 \land z = z_0 * 2\}\ \{y = z\}]$
$\{x > y_0 \land y = z\}$

6. Prove $\{x = y = z = c\}\ [\, x \coloneqq x^2 \parallel y \coloneqq y^2 \parallel z \coloneqq (z-d)*(z+d)]\ \{x = y = z + d^2\}$, where $c, d$ are named constants.
   - This is the same question as Example 1. But without the provided precondition in each thread, it is very natural to use $x = y = z = c$ as the precondition of every thread, then we have the following **wrong** partial proof outline if we use the parallelism rule:

     $\{x = y = z = c\}$
     $[\,\{x = y = z = c\}\ x \coloneqq\ x^2\ \{x = c^2 \wedge y = z = c\}$
     $\parallel \{x = y = z = c\}\ y \coloneqq y^2 \{y = c^2 \wedge x = z = c\}$
     $\parallel \{x = y = z = c\}\ z \coloneqq (z - d\,) * (z + d)\ \{z = c^2 - d^2 \wedge x = y = c\}]$
     $\{x = c^2 \wedge y = z = c \wedge y = c^2 \wedge x = z = c \wedge z = c^2 - d^2 \wedge x = y = c\}$

     It is wrong since interference in conditions happens everywhere. We need to age all the variables in preconditions.

   - For the first thread: $\{x_0 = y_0 = z_0 = c \wedge x = x_0\}\ x \coloneqq x^2\ \{x_0 = y_0 = z_0 = c \wedge x = x_0^2\}$.
   - For the second thread: $\{x_0 = y_0 = z_0 = c \wedge y = y_0\}\ y \coloneqq y^2\ \{x_0 = y_0 = z_0 = c \wedge y = y_0^2\}$.
   - For the third thread: $\{x_0 = y_0 = z_0 = c \wedge z = z_0\}\ z \coloneqq (z-d)*(z+d)\ \{x_0 = y_0 = z_0 = c \wedge z = z_0^2 - d^2\}$.

   - In the end, we get the following full proof outline:

     $\{x_0 = y_0 = z_0 = c \wedge x = x_0 \wedge y = y_0 \wedge z = z_0\}$
     $[\,\{x_0 = y_0 = z_0 = c \wedge x = x_0\}\ x \coloneqq x^2\ \{x_0 = y_0 = z_0 = c \wedge x = x_0^2\}$
     $\parallel \{x_0 = y_0 = z_0 = c \wedge y = y_0\}\ y \coloneqq y^2\ \{x_0 = y_0 = z_0 = c \wedge y = y_0^2\}$
     $\parallel \{x_0 = y_0 = z_0 = c \wedge z = z_0\}\ \ z \coloneqq (z-d)*(z+d)\ \{x_0 = y_0 = z_0 = c \wedge z = z_0^2 - d^2\}]$
     $\{x_0 = y_0 = z_0 = c \wedge x = x_0^2 \wedge y = y_0^2 \wedge z = z_0^2 - d^2\}$
     $\{x = y = z + d^2\}$

Observations about the Parallelism Rule

- So far, our requirements for using the parallelism rule is that the parallel program must be a disjoint parallel program, and it has disjoint conditions. Is this requirement too strict? Yes! What we really need for applying the parallelism rule is that "threads should not interfere with each other" or "interference free". Let's look at some examples and think about what is actually needed for "interference free".

7. Let's look at the following threads with their provable pre- and post- conditions, can we use parallelism rule to prove the parallel composition of them?

   a) $\{x = 0\}\ y \coloneqq y + 1\ \{x = 0\}$
      $\{y > 1\}\ x \coloneqq 0\ \{y > 1\}$

      Yes, $\{x = 0 \wedge y > 1\}\ [y \coloneqq y + 1 \parallel x \coloneqq 0]\ \{x = 0 \wedge y > 1\}$ is totally correct; even though these disjoint threads don't have disjoint conditions.

   b) $\{x = 0\}\ y \coloneqq y + 1\ \{x = 0\}$
      $\{z = 0\}\ y \coloneqq y + 2\ \{z = 0\}$

      Yes, $\{x = 0 \wedge z = 0\}\ [y \coloneqq y + 1 \parallel y \coloneqq y + 2]\ \{x = 0 \wedge z = 0\}$ is totally correct; even though these two threads are not disjoint.

c) $\{x > 0\}\ x \coloneqq x + 1\ \{x > 1\}$
$\{x > 0\}\ x \coloneqq x + 2\ \{x > 2\}$

Yes, $\{x > 0\}\ [x \coloneqq x + 1 \parallel x \coloneqq x + 2]\ \{x > 1 \land x > 2\}$ is totally correct; even though these two threads are not disjoint, and they don't have disjoint conditions.

- From the above examples, we can see that neither being disjoint, nor disjoint conditions is necessary for "interference free". Then what is necessary? Please think about it and we will discuss it in the next lecture.